

Archivos modificados:
Entry.S
Interrupt.c
Sched.c
Sys.c
Sys_call_table.S
System.c
Wrappers.S
Makefile
Errno.h

System.c

- main function
- das de donde se inicializa el sistema

variables globales

entry.S

- handlers
- entry point to the system

• toda funcion que queramos implementar tiene que estar aqui declarada

Siempre tiene que tener:

```
#define EOI
movl $0x20, %al
outb %al, $0x20

#define RESTORE_ALL
popl %ebx
popl %ecx
popl %edx
popl %esi
popl %edi
popl %ebp
popl %eax
popl %ds
popl %es
popl %fs
popl %gs
```

EOI para acabar la interrupción

• RESTORE_ALL con estos valores se refiere a la system stack de handlers

Escribir una handler:

```
SAVE_ALL
call routine
RESTORE_ALL
EOI
IRET
```

interrupt.c

- servicio d' interrupciones y excepciones

• se programan tanto las interrupciones como las rutinas

Rutinas cada una es diferente

figuran-se en el io.c

Interrupciones:

- position
- # handler
- prioLevel

set InterruptHandler (...)
↓
dentro del set Idt

añadir el errno.h para así tener datos los fallos

para poder ejecutar-se, es necesario los write MSR necesarios

Combin hardware.c última flag

Programar el write MSR i el syscall_handler_system

necesario para las interrupciones que queremos programar en el entry.S

lib.c

- para añadir funciones auxiliares

• pueden ser usados por el user.c

User.c

- lo que hace el user
- tiene que tener los privilegios necesarios

es lo que se muestra por pantalla

wrappers.S

- Assembler
- Para programar funciones

• Creamos el archivo

sys.c

- Añadir las funciones del wrapper en c

• Se llaman sys_funcion

utils.c

- funciones

• mas pueden ser utiles ya que mas ahorran trabajo

sys_call_table.S

- Cada entrada tiene una posición determinada

• hay que encontrar la posición correcta y daban el MAX_SYSCALL